EXHIBIT B

UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF MASSACHUSETTS

SINGULAR COMPUTING LLC, Plaintiff, v. GOOGLE LLC, Defendant.	Civil Action No. 1:19-cv-12551 FDS Hon. F. Dennis Saylor IV

RESPONSIVE CONTENTIONS REGARDING
NON-INFRINGEMENT AND INVALIDITY

Table of Contents

RESPONSIVE CONTENTIONS REGARDING NON-INFRINGEMENT AND INVALIDITY

Pursuant to the Scheduling Order (ECF No. 59) and Local Rule 16.6, Google LLC ("Google") hereby submits these Responsive Contentions Regarding Non-Infringement and Invalidity ("Responsive Contentions"). These disclosures, including the accompanying appendix and claim charts, set forth Google's responsive contentions with respect to the claims identified in the Infringement Contentions served by Plaintiff Singular Computing LLC ("Singular") on September 4, 2020 (the "Infringement Contentions," and the claims identified therein, the "asserted claims").

I. PRELIMINARY STATEMENT AND RESERVATIONS OF RIGHTS

Singular alleges infringement of the following asserted patents and asserted claims:

- U.S. Patent No. 8,407,273 (the '273 patent), claim 53;
- U.S. Patent No. 9,218,156 (the '156 patent), claim 7; and
- U.S. Patent No. 10,416,961 (the '961 patent), claims 4 and 13.

As set forth below, the accused TPUs do not infringe any asserted claim, and each asserted claim is invalid pursuant to 35 U.S.C. §§ 101, 102, 103, and/or 112.

These Responsive Contentions are based on information currently available to Google. Google's investigation and analysis of prior art is, however, ongoing. Furthermore, as set forth in related correspondence and meet-and-confer calls, Singular's Infringement Contentions are high-level and generally non-specific. Google thus reserves the right to supplement or modify these Responsive Contentions based on continued discovery, evaluation of the scope and content of the prior art, and/or changes in Singular's asserted claims or contentions.

In addition, because the Court has not yet issued a claim construction ruling, Google cannot provide complete and final invalidity contentions at this time. In the interim, Google's

Responsive Contentions may be based in part on the claim interpretations apparently underlying Singular's Infringement Contentions, to the extent that they are discernable. These Responsive Contentions are not intended to, and do not necessarily, reflect Google's positions as to the proper construction of the asserted claims. To the extent that the following Responsive Contentions reflect an interpretation consistent with the interpretation adopted by Singular's Infringement Contentions, no inference is intended nor should any be drawn that Google agrees with those claim constructions, and Google expressly reserves its right to contest such constructions. Further, no inference is intended nor should any be drawn that the claim limitations satisfy 35 U.S.C. § 112, and Google reserves the right to contend otherwise.

A. Priority Date

Singular "bears the burden of establishing that its claimed invention is entitled to an earlier priority date than an asserted prior art reference." *In re Magnum Oil Tools Int'l, Ltd.*, 829 F.3d 1364, 1376 (Fed. Cir. 2016). "To obtain the benefit of the filing date of a parent application, the claims of the later-filed application must be supported by the written description in the parent 'in sufficient detail that one skilled in the art can clearly conclude that the inventor invented the claimed invention as of the filing date sought.'" *Anascape, Ltd. v. Nintendo of Am., Inc.*, 601 F.3d 1333, 1335 (Fed. Cir. 2010) (quoting *Lockwood v. Am. Airlines, Inc.*, 107 F.3d 1565, 1572 (Fed. Cir. 1997)).

For the purposes of these Responsive Contentions, Google generally assumes that Singular's asserted priority date is June 19, 2009, the filing date of U.S. Provisional Application No. 61/218,691. But Google reserves the right to challenge any priority date and any alleged date of conception and to amend these contentions upon the Court's claim construction order, the Court's findings concerning the priority date(s) of the asserted claims, information learned through discovery, or otherwise. Specifically, as discussed in more detail below and set forth in

its related petitions for *inter partes* review (Case Nos. IPR2021-00154, IPR2021-00164, and IPR2021-00178), which Google incorporates by reference, U.S. Patent Application No. 12/816,201 (U.S. Patent No. 8,150,902) as well as U.S. Provisional Application No. 61/218,691 lacks written description and enabling disclosures; therefore, Singular is not entitled to the priority date of either or those applications. Instead, the applicable filing dates should be the actual filing dates of the applications that led to the '273, '156, and '961 patents.

II. NON-INFRINGEMENT

Singular's Infringement Contentions fail to demonstrate a plausible basis on which Singular can carry its burden of proof to show that the accused products that Singular has identified meet each of the asserted claims. In fact, Singular's contentions fail to demonstrate a plausible basis on which Singular can carry its burden even as to *any* single element of the claims. Singular fails to identify what it contends in the accused products is the basic building block on which the claims are built: the low-precision high dynamic range execution unit (LPHDR execution unit). Without that building block, its contentions fail to demonstrate a plausible basis for showing the rest of the claim elements can be met as well. Furthermore, as to various elements of each claim, Singular's contentions fail to show a plausible basis on which the claim element can be satisfied for reasons independent of the failure to identify a LPHDR execution unit. Fundamentally, the problem with Singular's Infringement Contentions is that they are attempting to identify something that does not exist: any part of the TPU that performs an error-tolerant, approximate arithmetic operation. To the contrary, the accused TPUs perform only exact mathematical operations, including in the MXU.

Pursuant to the Scheduling Order (ECF No. 59) and Local Rule 16.6(d)(4)(D), the attached Exhibit 18 provides further detail, on a claim-by-claim basis, of Google's response to Singular's Infringement Contentions.

Discovery is ongoing, claim construction is not yet completed, and Singular's production of certain materials relied on its Infringement Contentions was delayed; therefore, Google reserves the right to amend and/or supplement these non-infringement contentions.

III. NON-PATENTABLE SUBJECT MATTER UNDER 35 U.S.C. § 101

Google's asserted bases for invalidity under 35 U.S.C. § 101 are based on Google's investigation thus far. Google reserves its right to amend or otherwise modify its asserted bases for invalidity under 35 U.S.C. § 101 based on its review of additional documents and evidence, including, without limitation, additional information concerning the state of art and level of one of ordinary skill in the art at the relevant time. Further, Google reserves the right to assert additional arguments of invalidity under 35 U.S.C. § 101 based on the claim construction process, Singular's proposed claim constructions, any supplement to Singular's infringement contentions, Singular's expert reports, or other positions taken by Singular. A more detailed basis for § 101 defenses is set forth in Google's Memorandum in Support of its Motion to Dismiss, *see* ECF No. 41, and will be set forth in Google's expert reports and/or in pleadings.

The asserted claims are not patent-eligible under 35 U.S.C. § 101 because they claim abstract ideas. In particular, the asserted claims are generally directed to an abstract concept, namely, performing mathematical calculations with some intended amount of imprecision using a number representation with high dynamic range. Indeed, as discussed in more detail below, Singular appears to contend that its claims encompass any type of processor that operates on a particular subset of numbers (e.g., floating point numbers with some maximum amount of precision bits and minimum amount of exponent bits), which would not be eligible for patenting. *See Mackay Co. v. Radio Corp.,* 306 U.S. 86 (1939) (noting that "a scientific truth, or the mathematical expression of it, is not patentable"); *see also Gottschalk v. Benson*, 409 U. S. 63 (1972); *Parker v. Flook*, 437 U.S. 584 (1978); *Diamond v. Diehr*, 450 U. S. 175, 185 (1981)

(holding that "a mathematical formula, like a law of nature, cannot be the subject of a patent."); *Bilski v. Kappos*, 561 U. S. 593, 611 (2010) ("*Diehr* explained that . . . an abstract idea, law of nature, or mathematical formula could not be patented[.]'"); *SAP America v. Investpic LLC*, 898 F. 3d 1161, 1166 (Fed. Cir. 2018) (stating that "mathematical calculations and formulas are not patent eligible."). Furthermore, the patents merely recite implementing this abstract idea on some type of hardware device, without any attempt to offer an inventive insight or improvement. But it is beyond question that the recitation of a generic computer implementation or a particular technological environment cannot transform an abstract idea into patent-eligible subject matter. *See, e.g., Alice Corp. v. CLS Bank Int'l*, 134 S. Ct. 2347, 2355 (2014); *Tech LLC v. BuySeasons, Inc*., 899 F.3d 1281, 1290 (Fed. Cir. 2018). Accordingly, the asserted claims are invalid.

IV. PRIOR-ART-BASED INVALIDITY

A. Identification of Prior Art

The asserted patents share a substantially identical specification and have similar claims. Below, Google provides a consolidated list identifying prior art that anticipates and/or renders obvious one or more claims of the asserted patents under at least one of 35 U.S.C. §§ 102(a), (b), (e), or (g) and/or 35 U.S.C. § 103, including relevant dates where presently known. In these Responsive Contentions, including the claim charts, any citation to a printed publication or other reference describing a prior art system should also be construed to include a reference to the prior art system itself. Each listed document or item became prior art at least as early as the dates set forth herein. Google reserves the right to rely upon any systems, products, or prior inventions related to any of the references identified in these Responsive Contentions.

Patent or Application Number	Date of Filing, Issuance, and/or Publication
U.S. Patent No. 5,892,962 ("Cloutier")	April 6, 1999

U.S. Patent No. 5,442,577 ("Cohen")	August 15, 1995
U.S. Patent App. Publ. No. 2007/0203967 ("Dockser")	August 30, 2007
U.S. Patent Appl. Publ. No. 2009/0066164 ("Flynn")	March 12, 2009
U.S. Patent No. 5,666,071 ("Hawkins")	September 9, 1997
U.S. Patent No. 5,689,677 ("MacMillan")	November 18, 1997
U.S. Patent No. 6,311,282 ("Nelson")	October 30, 2001
U.S. Patent Appl. Pub. No. 2003/0204750 ("Youngs")	October 30, 2003

Prior Art References	**Date**
Adaptive Solutions Connected Network of Adaptive Processors (CNAPS) Neurocomputer Chip ("CNAPS")	1991
Arnold et. al, "Splash 2" *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures* ("Splash 2")	June 1992
Aty et al, *High-Speed, Area-Efficient FPGA-Based Floating-Point Multiplier* ("Aty")	December 9-11, 2003
Belanović, *Library of Parameterized Hardware Modules for Floating-Point Arithmetic with an Example Application* ("Belanović")	May 2002
Belanović and Leeser, *Library of Parameterized Floating-Point Modules and Their Use* ("Belanović and Leeser")	2002
Cray T3D System ("Cray T3D")	1994
Hoefflinger et. al, *Digital Logarithmic CMOS Multiplier for Very-High-Speed Signal Processing* ("Hoefflinger")	1991
Intel x86 Microprocessor Architecture, including the 80386 microprocessor and 80387 co-processor and the 80486 microprocessor	1985/1989
Intel i860 Microprocessor Architecture, including the iPSC/860 massively parallel supercomputer system	1990
Lee et al, *An FPGA-Based Face Detector Using Neural Network and a Scalable Floating Point Unit*, Proceedings of the 5th WSEAS	2006

International Conference on Circuits, Systems, Electronics, Control & Signal Processing at 315-320 ("Lee")	
Makino, *Grape and Project Milkyway*, Proceedings of the 6th East Asian Meeting of Astronomy	October 18-22, 2004
Makino et al, *GRAPE-6: Massively-Parallel Special-Purpose Computer for Astrophysical Particle Simulation*s, Astronomical Society of Japan	1993
MANTRA I	1993
Okumura et al, *GRAPE-3: Highly Parallelized Special-Purpose Computer for Gravitational Many-body Simulations* ("GRAPE-3")	1992
Okumura et al, *Highly Parallelized Special-Purpose Computer, GRAPE-3*, Astronomical Society of Japan	1992
Shirazi et al, *Quantitative Analysis of Floating Point Arithmetic on FPGA Based Custom Computing Machines* ("Shirazi")	April 1995
SPERT-II	1995
Sudha et al, *An Efficient Digital Architecture for Principal Component Neural Network and its FPGA Implementation* ("Sudha")	2007
Tong et al, *Reducing Power by Optimizing the Necessary Precision/Range of Floating-Point Arithmetic* ("Tong")	June 2000
SYNAPSE-1	1993
Texas Instruments TMS320C32 DSP	1995
Xilinx Virtex-4 FPGA	2007

B. Anticipation

Google contends that the following prior art anticipates the asserted claims of the patents-in-suit under 35 U.S.C. § 102:

- Belanović / Belanović and Leeser
- GRAPE-3

- Cray T3D

Google further contends that the following prior art anticipates the asserted claims of the '961 patent under 35 U.S.C. § 102:

- Lee
- Sudha
- Dockser
- Shirazi
- Aty
- TMS320C32
- Hoefflinger

The attached respective claim charts (Exhibits 1-17) identify how the prior art discloses each limitation of each asserted claim. The charts are exemplary only and representative of the content and teaching of the prior art; they should be understood in the context of each prior art reference as a whole as it would be understood by a person of ordinary skill in the art. Google reserves the right to rely on other disclosures, particularly if Singular contests the scope and content of the disclosures identified in the claim charts and/or the knowledge of one of ordinary skill in the art.

Google's invalidity theories may depend on the Court's claim constructions. In addition, Google's Responsive Contentions may also reflect implicit positions advanced by Singular regarding the interpretation of any asserted claim or particular limitation. The contentions disclosed herein do not constitute and should not be taken as any endorsement, acquiescence, or acceptance by Google of any interpretation advanced by Singular. Google reserves the right to

amend these Responsive Contentions pending further developments in this case, including as to claim construction and any amendments to Singular's Infringement Contentions.

C. Obviousness

Even if not identically disclosed, a patent claim may be invalid if the differences between the claimed invention and the prior art are such that the claimed invention as a whole would have been obvious before the effective filing date of the claimed invention to a person having ordinary skill in the art to which the claimed invention pertains. The legal determination of obviousness is based on underlying factual questions set forth in *Graham v. John Deere Co. of Kansas City*:

1. the scope and content of the prior art;
2. differences between the prior art and the claims at issue;
3. the level of ordinary skill in the pertinent art; and
4. evaluation of any relevant secondary considerations.

383 U.S. 1, 17 (1966).

As the U.S. Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 127 S. Ct. 1727, 1739 (2007). The Supreme Court further held that, "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 1740.

Moreover, the Supreme Court held that "in many cases a person of ordinary skill will be able to fit the teachings of multiple patents together like pieces of a puzzle." *Id.* at 1742. Indeed,

the Supreme Court held that it is sufficient that a combination of elements was "obvious to try" holding that, "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.* "In that instance the fact that a combination was obvious to try might show that it was obvious under § 103." *Id.* "Rigid preventative rules that deny factfinders recourse to common sense, however, are neither necessary under our case law nor consistent with it." *Id.*

Finally, the Supreme Court recognized that "[g]ranting patent protection to advances that would occur in the ordinary course without real innovation retards progress and may, in the case of patents combining previously known elements, deprive prior inventions of their value or utility." *Id.* at 1741.

For the asserted patents, each of the above-identified prior art items is directed to the same or similar fields of endeavor. Accordingly, one of ordinary skill in the art would have recognized that the results of the combinations were predictable, and would have been clearly motivated to modify and combine the prior art items identified above to arrive at the alleged inventions of claims of the asserted patents. At least the following rationales support a finding of obviousness based on the combinations set forth in detail below:

(A) Combining prior art elements according to known methods to yield predictable results;

(B) Simple substitution of one known element for another to obtain predictable results;

(C) Use of known technique to improve similar devices (methods, or products) in the same way;

(D) Applying a known technique to a known device (method, or product) ready for improvement to yield predictable results;

(E) "Obvious to try"—choosing from a finite number of identified, predictable

solutions, with a reasonable expectation of success;

(F) Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations would have been predictable to one of ordinary skill in the art; and

(G) Some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

Evidence of contemporaneous invention may also support a ruling that a claimed invention was obvious to those of skill in the art. *See, e.g., Concrete Appliances Co. v. Gomery*, 269 U.S. 177, 185 (1925); *Geo M. Martin Co. v. Alliance Machine Sys. Int'l LLC*, 618 F.3d 1294, 1305–06 (Fed. Cir. 2010); *Senju Pharm. Co. Ltd. v. Apotex Inc.*, 717 F. Supp. 2d 404, 425-26 (D. Del. 2010). To the extent that any of the references in the claim charts are not found to anticipate the asserted claims, Google contends that those claims are invalid under 35 U.S.C. § 103.

1. Prior Art Background

a. Floating-point format and arithmetic

Dr. Bates did not invent the floating-point format, which is used to express a higher dynamic range of numbers. That format, which simply applies pre-computing concepts of numeric representation, dates back to the earliest days of computing, including the Z3 machine developed by Konrad Zuse. *See* Raúl Rojas, *Konrad Zuse's Legacy: The Architecture of the Z1 and Z3*, IEEE Annals of the History of Computing 19.2 at 5-16 (1997). Since then, the format has become ubiquitous, so much so that it is the subject of an industry standard. Specifically, IEEE 754 was first adopted in 1985 and officially updated in 2008 (and 2019).

The floating-point format is commonly taught in discussing computer architecture and design. *See, e.g.*, Patterson and Hennessy, *Computer Organization and Design* § 4.8 (2d. ed. 1998). What specific floating-point format to use is application-dependent and involves balancing the desired precision and range: "The designer of a floating-point representation must

find a compromise between the size of the significand and the size of the exponent, because a fixed word size means you must take a bit from one to add a bit to the other. This trade-off is between accuracy and range: Increasing the size of the significand enhances the accuracy of the significand, while increasing the size of the exponent increases the range of numbers that can be represented." *Id.* at 276. Given the ubiquity of floating-point formats, it was also well known in the art how to design circuits for floating-point arithmetic, including for floating-point addition and multiplication. *See id.* at 285 (block diagram of circuit for floating-point addition), 289 (flow chart for floating-point multiplication).

b. Reduced-precision floating-point formats

There are various industry standards that use low-precision and/or high dynamic range floating-point formats (as compared to the standard 32-bit floating-point format). For example, the OpenEXR standard developed by Industrial Light and Magic used a 16-bit "half" floating-point format with 1 sign bit, 10 mantissa bits, and 5 exponent bits. *See* https://www.openexr.com/about.html. The OpenGL standard used a similar format. *See generally* https://www.opengl.org/. In 2004, NVIDIA introduced its "half" or "float16" format. https://www.khronos.org/registry/OpenGL/extensions/NV/NV_half_float.txt. As explained at the time: "This extension introduces a new storage format and data type for half-precision (16-bit) floating-point quantities. The floating-point format is very similar to the IEEE single-precision floating-point standard, except that it has only 5 exponent bits and 10 mantissa bits. Half-precision floats are *smaller than full precision floats and provide a larger dynamic range than similarly-sized normalized scalar data types*." *Id.* (emphasis added). The OpenGL standard also supports other low-bitdepth floats besides the "half" format, including R11F_G11F_B10F, which uses 11- and 10-bit floats with 5-bit exponents and, respectively, 6- and 5-bit mantissas. *See* https://www.khronos.org/opengl/wiki/Small_Float_Formats. And OpenGL supported the

RGB9_E5 format, which has a 9-bit mantissa and 5-bit exponent. *See id.* Notably, the asserted patents' common specification acknowledges that these formats existed before Dr. Bates' alleged invention. '273 patent at 5:11-30.

Notably, the Z3 computer, discussed above, also had high dynamic range and lower precision than the later-developed IEEE 754 standard for single precision floating point format; specifically, it had an 8-bit exponent (like the IEEE standard) and only 14 bits for the significand (compared to the 23 fraction bits for the IEEE standard). *See* Raúl Rojas *Konrad Zuse's Legacy: The Architecture of the Z1 and Z3*, IEEE Annals of the History of Computing 19.2 at 6.

c. Low-precision arithmetic for machine learning

Relying on low-precision calculations for machine learning was not Dr. Bates' idea, either. Those skilled in the art touted the benefits of using low-precision math for machine learning long before Dr. Bates' purported invention. As Dr. Asanović and other computer science professors described in 1992, it was "commonly held that ANN [artificial neural network] calculations [did] not carry the high-precision requirement of mainstream scientific applications." Asanović et. al, *Using Simulations of Reduced Precision Arithmetic to Design a Neuro-Microprossesor*, at 33-34. "[I]n contrast with scientific applications," they explained, "artificial neural network back-propagation training algorithms work well with moderate precision arithmetic." *Id.* at 34. Furthermore, as Dr. Asanović demonstrated in the context of a large speech classification application, using reduced precision comparable to "32b floating point" was preferable because it avoided the "inefficiencies" inherent in more precise calculations. *Id.* In particular, he noted, "excess precision operands waste valuable processor-memory bandwidth." *Id.*

Expanding upon his earlier research, in 2002, Dr. Asanović said that "limited numeric precision" had become one of "the primary distinguishing characteristics" of neurocomputers, or

machines specifically programmed for ANN computations. Asanović, *Programmable Neurocomputing*, The Handbook of Brain Theory and Neural Networks, 2nd ed., at 6. He explained that reduced precision in neurocomputers was preferable to more precise calculations, because it "allow[ed] reductions in the area required for arithmetic circuits, *particularly multipliers*, and also reduce[d] the bandwidth required to transfer operands." *Id.* at 4 (emphasis added). As he had recognized with his colleagues a decade earlier, reduced precision calculations were thus well-suited for machine learning applications.

2. Differences Between the Prior Art and Asserted Claims

The asserted patents claim a processor or other device for performing "low-precision, high-dynamic range" arithmetic. But Dr. Bates himself acknowledged that low precision high dynamic range arithmetic was not his idea. As he "freely admits," "[i]mperfect computing . . . is not an idea that's unique to his startup company, Singular Computing." *Imperfect Processing: A Functionally Feasible (and Fiscally Attractive) Option, Says Singular Computing*, Berkeley Design Technology Inc., Oct. 23, 2013, at 1. Indeed, he acknowledged that several "large famous companies" have explored approximation arithmetic to reduce transistor count and power consumption. *Id.* Even in Dr. Bates' view, the only apparently novel aspect of Singular's method was implementing low precision high dynamic range arithmetic on an "approximate processing element." *Id.* In Dr. Bates' words—and in keeping with the discussion above—"everything else about [Singular's] machine you can read about in a 1994 textbook." *Id.* at 2.

As discussed in further detail below, to the extent that certain of the prior art did not disclose all of the elements of the asserted claims, various combinations would have rendered those claims obvious.

3. Level of Ordinary Skill in the Art

A person of skill in the art in mid-2008 would have had at least a bachelor's degree in Electrical Engineering, Computer Engineering, Applied Mathematics, or the equivalent, and at least two years of academic or industry experience in computer architecture. More education could substitute for less experience, and vice versa.

As noted above, Google believes none of the asserted claims are entitled to any of the claimed earlier priority date(s); therefore, the timeframe for evaluating obviousness of the asserted claims is, respectively, the actual filing date for each asserted patent. In any event, the combinations detailed below would have been obvious to one of skill in the art even at the time of the provisional application underlying the asserted patents. That is, irrespective of which priority date one uses, the asserted claims would have been obvious to one of skill in the art.

4. Secondary Considerations of Non-Obviousness

Singular bears the burden of identifying any objective indicia of non-obviousness. *See, e.g.*, *In re Dillon*, 919 F.2d 688, 692 (Fed. Cir. 1990) (en banc) (holding that if a *prima facie* case of obviousness is established, the burden shifts to the applicant to come forward with arguments and/or evidence to rebut the prima facie case). Google therefore reserves the right to address and/or challenge any such objective criteria if and when identified by Singular.

D. Detailed Discussion of Anticipation and Obviousness Grounds

Each of the separate elements of the asserted claims was well known to those of the ordinary skill in the art before the alleged inventions. The "inventions" recited in the asserted claims were either already disclosed in a single prior art system or publication or, at the very most, reflected the simple combination of well-known elements with known methods to yield predictable results. These well-known elements and methods include field-programmable gate arrays with custom floating-point formats; LPHDR execution units using logarithmic

representations; devices that reduce precision as part of floating-point operations; processors capable of operating on reduced-precision floating-point numbers; systems supporting variable precision floating-point numbers and/or operations; processors with reduced-precision floating-point number formats; low-precision ANN systems using fixed-point formats; and systolic arrays of numerous processing elements. A person of ordinary skill in the art would be able and motivated to combine these individual, known elements in the art to yield predictable results.

Examples of the prior public use and disclosure of some of these elements are provided below. Some implementations of these elements alone are enough to render every claim in the asserted patents obvious. *See, e.g.,* Pavle Belanović, *Library of Parameterized Hardware Modules for Floating-Point Arithmetic with an Example Application* (2003). Additional disclosures are included in the attached claim charts, which are incorporated herein by reference.

1. Field-Programmable Gate Arrays with Custom Floating-Point Formats

Dr. Bates purports to have invented FPGAs with custom floating-point formats, specifically floating-point formats that allow multiplication to occur with low precision and a high dynamic range. In the common specification, Dr. Bates alleges that "modern FPGAs often devote a significant portion of their area to providing dozens or hundreds of multiplier blocks, which can be used instead of general purpose resources for computations requiring multiplication." *See, e.g.,* '273 patent at 4:46-49. He then dismisses these "modern FPGAs" because their "multiplier blocks typically perform 18 bit or wider integer multiplies." *Id.* at 4:46-52. Dr. Bates' analysis relies on a flawed belief that FPGAs prior to his alleged inventions typically performed fixed-point, integer multiplication in an 18-bit format or higher.

As floating-point capability became more affordable in the 1990s, applications that "traditionally . . . used integer math turn[ed] to floating-point representation." Geir Kjosavik,

Tutorial: Floating-point arithmetic on FPGAs (2006), at 3, accessible at https://www.eetimes.com/tutorial-floating-point-arithmetic-on-fpgas/#. Although those skilled in the art were already experimenting with floating-point arithmetic on FPGAs, "[d]esign and implementation of floating point arithmetic and mapping of this into an FPGA became easier with the emergence of new generation FPGAs and development of high level languages such as VHDL tools." Sahin S., Kavak A., Becerikli Y., Demiray H.E., *Implementation of floating point arithmetics using an FPGA* (2007), accessible at https://doi.org/10.1007/978-1-4020-5678-9_39.

As of the priority dates of the asserted patents, those of skill in the art knew they could use FPGAs with custom floating-point formats, including 16-bit formats designed for LPHDR operations. As early as 1995, "Quantitative Analysis of Floating Point Arithmetic on FPGA Based Custom Computing Machines" recognized that 16-bit floating-point formats on FPGAs were "designed as a compromise between data width and a large enough dynamic number range." Shirazi et al., *Quantitative Analysis of Floating Point Arithmetic*, at 2. The authors recognized that "custom formats, derived for individual applications, are feasible on CCMs, and can be implemented on a fraction of a single FPGA." *Id.* at 1; *see also* Gaffar et al., *Unifying Bit-width Optimisation for Fixed-point and Floating-point Designs,* at 4 (2004) (floating-point "precision depends on the mantissa bit-width, while the range depends on the exponent bit-width"; "customised arithmetic formats, where we can have arbitrary integer and fractional widths for fixed-point designs, and arbitrary mantissa and exponent widths for floating-point designs" allow for low precision and high dynamic range). *Id.* at 3. Shirazi and his co-authors implemented their custom floating-point formats on a Splash-2 custom computing machine, which used multiple Xilinx 4010 FPGAs. *See, e.g.,* Arnold, Buell, and Kleinfelder, "Splash 2," *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, June

1992 (describing the Splash 2 architecture as used with the Sun SPARC 2 workstations); Arnold, Buell, and Davis, "Splash 2," *ACM Symposium on Parallelism in Algorithms and Architectures* (1992) (same). The disclosures of Shirazi, including its use of the Splash-2 architecture, are detailed in Exhibit 8.

Later references that similarly predate the asserted patents also disclose the use of FPGAs with custom floating-point formats. For example, in 2003, "High-Speed, Area-Efficient FPGA-Based Floating-Point Multiplier" described a floating-point multiplier that had been designed, optimized, and implemented on an FPGA based system. *See* Aty et al., *High-Speed, Area-Efficient FPGA-Based Floating-Point Multiplier* (2003). Using a 16-bit format as an example, Aty and his co-authors described how "[s]calable floating-point multiplier allows manipulating range and precision . . . by controlling the size of mantissa and exponent fields." *Id.* at 274, 276 (providing a chart regarding the "optimized result" for 16-bit floating point multipliers on an FPGA). Moreover, in 2007, "An Efficient Digital Architecture for Principal Component Neural Network and its FPGA Implementation" disclosed a 64-input, 16-output pulse coupled neural network. Sudha et al., *An Efficient Digital Architecture for Principal Component Neural Network and its FPGA Implementation* (2007). Each parameter in Sudha's hardware design was "represented as a floating point number with an 8-bit mantissa and an 8-bit exponent." *Id.* at 428; *see also* Appendix (detailing how an 8-8 floating point number meets the error rates and dynamic range allegedly disclosed by Dr. Bates). One of skill in the art could easily have implemented Sudha using a more recent FPGA, or--similar to Belanović and Leeser (discussed below)--could simply have used multiple Xilinx FPGAs. The disclosures of Aty and Sudha are detailed in Exhibits 9 and 12.

As another example, in 2002, Pavle Belanović and Miriam Leeser noted what was, by then, an inherent concept in computing: "[a] natural tradeoff exists between smaller bitwidths requiring fewer hardware resources and higher bitwidths providing better precision. Also, within a given total bitwidth, it is possible to assign various combinations of bitwidths to the exponent and fraction fields, where wider exponents result in higher range and wider fractions result in better precision." Belanović & Leeser, *A Library of Parameterized Floating-Point Modules and Their Use*, at 658 (relying on Shirazi et al. as "significant work" in advancing the use of custom floating-point formats). Belanović and Leeser recognized that custom floating-point formats were an inevitable outgrowth of such an understanding, because "[r]educed bitwidth implementations require fewer resources and thus allow for more parallel implementations." *Id.* Using the Wildstar reconfigurable architecture (based on Splash 2) and using a Xilinx XCV1000 FPGA, Belanović and Leeser implemented custom floating-point formats. *Id.* at 663 (describing multiple 16-bit and 12-bit formats that fall within Dr. Bates' disclosed precision and dynamic ranges). They concluded that "when using floating-point arithmetic, the designer has full control to trade off between range and precision." *Id.* at 666. Years before Dr. Bates disclosed his alleged inventions, those skilled in the art were mixing and matching floating-point formats depending on application-specific needs and implementing those formats using reconfigurable FPGAs. And years before Dr. Bates disclosed his alleged inventions, Belanović and Leeser made a parameterized floating-point library for use with reconfigurable hardware, a library which has since been publicly available for anyone to implement custom formats for FPGAs. The disclosures of Belanović and Leeser are detailed in Exhibit 10.

Also in 2002 (and still preceding the priority date of the asserted patents), Belanović applied the custom floating-point format in experiments using the Wildstar architecture and

Xilinx VIRTEX XCV1000 FPGAs, implementing a system that included 255 multiplication execution units. *See* Pavle Belanović, *Library of Parameterized Hardware Modules for Floating-Point Arithmetic with an Example Application* (2003). "Floating-point formats used in the experiments were chosen to represent the range of realistic floating-point formats from 8 to 32 bits," *id.* at 46, including 12-bit and 16-bit formats that meet Dr. Bates' error rates and dynamic range. Belanović chose custom floating-point formats implemented through an FPGA for the same reason Dr. Bates later describes in his alleged inventions: "When using floating-point arithmetic, the designer using the library has full control to trade off between range and precision. . . . With a wider exponent field, the designer provides larger range to the signal, while sacrificing precision. Similarly, to increase the precision of a signal at the cost of reduced range, the designer chooses a narrower exponent and wider fraction field." Belanović, *Library of Parameterized Hardware* at 50. "Reducing datapath bitwidths to their optimal values enables design of more parallel architectures and implementation of larger designs." *Id.* at 13. Belanović explained that his thesis bridged the gap between custom formats and "the required implementation support to enable the complete transition from algorithm to hardware." *Id.* at 70. His thesis provided "a complete solution" to implementation problems for custom formats, allowing new "possibilities for area and power savings." *Id.* And Belanović, like others in the art, recognized that applications of this research included "algorithms that are highly parallel and have signal values that have a high enough range to require floating-point representation, yet are tolerant enough to accommodate its lower precision." *Id.* at 69-70. Belanović noted that parallel processing was the obvious conclusion of his custom format research: "the range and precision required will ideally necessitate bitwidths significantly lower than those of the IEEE formats, so

that higher parallelism may be achieved." *Id.* at 70. The disclosures of Belanović are detailed in Exhibit 11.

Three years before the priority date of the asserted patents, in 2006, Lee and Ko developed another FPGA-based system utilizing reduced 16-bit precision for use in a face-detecting neural network. *See* Yongsoon Lee and Seok-Bum Ko, *An FPGA-Based Face Detector Using Neural Network and a Scalable Floating Point Unit* (2006) ("Lee"). They recognized that the use of floating point units "provides dynamic range and reduces the bit of the arithmetic unit more than fixed point method does. These features led to reduction in the memory so that it is efficient for neural networks system with large size data bits." *Id.* at 315. Relatedly, smaller bit sizes require significantly less area on a chip. *Id.* at 315, 317, 319. The cost of these efficiency and area gains, they noted, is decreased precision. But for use cases in neural networks, such as the face detector they developed, the lack of precision had minimal impact on the detection rate. *Id*. at 315, 319. Lee concluded that the number of bits used in a system's floating point units was customizable to the operation: "we need to decide the least number of bits within the acceptable error range." *Id.* at 317. The disclosures of Lee are detailed in Exhibit 17.

Moreover, FPGA-based parallel processing with more than 100 processors was known to those skilled in the art since at least 1999. The Cloutier patent (U.S. 5,892,962; issued Apr. 6, 1999) discloses an "FPGA-based" "multiprocessor comprising a multidimensional array of FPGAs" where the array is "adapted to be programmed as one or more processing elements (PEs)." Cloutier, Title, 1:47-53. "Depending on the application, each PE may correspond to one or more programmed FPGAs, or, alternatively, each FPGA may be programmed to operate as one or more PEs." *Id*., 3:5-8. Figure 3 of the patent shows an example wherein the FPGAs are programmed as 16 PEs and the multiprocessor "can be configured with one or more other such

multiprocessors 100 to form a large, more powerful parallel processing architecture." *Id*., 4:12-14. Cloutier's multiprocessor "provide[s] improved apparatus for data processing applications requiring a large number of operations, such as image processing, pattern recognition, and neural network algorithms." *Id*., 1:32-35. Over a decade before Dr. Bates disclosed his alleged inventions. Cloutier discussed the obvious concept: that "[i]ncreasing the number of PEs increases performance." *Id*., 5:36-37.

Based on the disclosures identified in Exhibits 8-13 and Exhibit 17, these references related to FPGAs implementing reduced precision floating point numbers, either alone or in combination with prior art regarding massively parallel processing (including Belanović and Cloutier, discussed above, and MacMillan and MANTRA I, discussed below), render the asserted claims invalid as obvious.

2. LPHDR Execution Units Using Logarithmic Representations

Dr. Bates purports to have invented a logarithmic embodiment of his LPHDR execution unit. *See* '273 patent at 6:10-22 (discussing logarithmic embodiment). In fact, in the common specification, Dr. Bates characterizes this embodiment as superior to the embodiment using a floating-point format, which the common specification says requires relatively large circuits for multiplication and division (circuits like the multiplier depicted above). *See id.* at 6:3-10 (discussing floating-point embodiment). Accordingly, the logarithmic embodiment is presumably encompassed within the asserted claims.

As of the priority date(s) of the asserted patents, however, using LPHDR execution units to perform arithmetic operations on logarithmic representations was known to those of skill in the art and, therefore, the asserted claims were either not novel or obvious. Indeed, these ideas are nearly as old as the computer itself. *See* Konrad Zuse, *Rechenmaschine*, 1949, available at ttp://zuse.zib.de/item/9MCS57vAMFIA2RVJ (1949 patent application describing the use of

logarithms instead of fixed or floating point; notably, Dr. Bates uses the same name for the auxiliary function as Zuse did 60 years before: "F").

More recent references that also precede the priority date(s) of the asserted patents also disclose the use of LPHDR execution units to perform arithmetic operations on logarithmic representations. This includes the GRAPE-3 computer, developed by S.K. Okumura and others at the University of Tokyo. The system described in their 1992 paper, "GRAPE-3: Highly Parallelized Special-purpose Computer for Gravitational Many-body Simulations," related to a special-purpose architecture for solving astronomical problems. GRAPE-3 had hundreds of arithmetic units in a chip, using LNS to allow for a greater utilization of the total transistor count than circuits designed for floating-point arithmetic. *See also*, Junichiro Makino, *GRAPE and Project Milkyway*, 2005 (summarizing the history of GRAPE). This same motivation was later adopted by Dr. Bates. *See* '273 Patent at 5:43-46 ("designers of traditional processors also have been struggling to use the enormous increase in available transistors to improve performance of their machines"). And subsequent work by Gerhard Lienhart and others at the University of Mannheim, published in 2002, showed that the GRAPE approach could be adapted to FPGA-based machines. *See* Gerhard Lienhart, Andreas Kugel and Reinhard Männer, "Using Floating-Point Arithmetic on FPGAs to Accelerate Scientific N-Body Simulations." The disclosures related to GRAPE-3 are detailed in Exhibit 15.

As another example, "Digital Logarithmic CMOS Multiplier for Very-High-Speed Signal Processing," published in 1991 by Hoefflinger, et al, also utilized "massively parallel systems" and LNS to dramatically reduce the transistor count and multiplication time needed for standard multipliers. Similarly, vanDrunen described a "novel arithmetic unit that can do computations in reduced precision," "speed[ing] up the process of simulation, while at the same time NOT

affecting the overall validity." *Arithmetic for Relative Accuracy* at 241. This novel arithmetic unit achieves these efficiency gains, according to vanDrunen, by introducing "parallelization" and "the use of binary logarithmic arithmetic." *Id.* The disclosures of Hoefflinger are detailed in Exhibit 16.

Based on the disclosures identified in Exhibits 15-16, GRAPE-3 and Hoefflinger, either standing alone or combination with prior art related to controllers and parallel processing (including Belanović and Cloutier, discussed above, and MacMillan and MANTRA I, discussed below) render the asserted claims invalid as obvious.

3. Devices That Reduce Precision as Part of Floating-Point Operations

People of skill in the art recognized that reduced precision floating point calculations were well-suited to a wide range of applications and yielded significant benefits in terms of reduced power consumption. As early as 2000, a group of computer science professors had demonstrated that certain applications, like voice recognition, image processing, and other signal-processing, could function properly using low precision high dynamic range arithmetic. Tong, *Reducing Power by Optimizing the Necessary Precision/Range of Floating-Point Arithmetic* (2000) at 273. As they explained, reducing precision "significantly reduce[d] [the floating point hardware's] power consumption," but caused almost no loss of overall accuracy in the results of various models. *Id.* at 274. Given that floating-point hardware was "very power hungry" and tended to be among the most "expensive components in a processor's power budget," those skilled in the art would have been motivated to reduce costs by implementing low-precision arithmetic on various types of floating point hardware. *Id.* at 273. And, as Tong demonstrated, doing so would have little downside, as reduced precision calculations had little impact on a program's overall accuracy. The disclosures of Tong are detailed in Exhibit 6.

Dockser, published in 2007, disclosed an execution unit that operated on numbers with high dynamic range but saved power by reducing unnecessary precision, with that reduction taking place in the operation of the floating point execution unit either by turning off the power to certain registers or in the course of the arithmetic operation, again by turning off power at an appropriate stage. Dockser at [0003]–[0007]. Dockser's "floating-point processor (FPP)" reduces precision below the IEEE-754 32-bit single format in which it receives input numbers by removing power from some of the least significant mantissa bits in the FPP's operation. *Id.* at [0015]–[0018], [0026]–[0027]. "By selecting the subprecision of the floating-point format" by removing power to least-significant mantissa bits, Dockser "reduc[es] the power consumption of the floating-point processor to support the selected subprecision." *Id.* at [0014]. The disclosures of Dockser are detailed in Exhibit 7.

As discussed above, before the invention, skilled artisans were well aware that reduced precision calculations were acceptable in a variety of applications, and they had reason to implement such calculations given that doing so would reduce power consumption without marked accuracy sacrifices. *See* Asanović, *Using Simulations of Reduced Precision Arithmetic to Design a Neuro-Microprossesor*, at 34 ("using reduced precision" was preferable because "excess precision operands waste valuable processor-memory bandwidth."); Patterson *et al*, *Computer Organization & Design*, at 276 (1998) ("Increasing the size of the significand enhances the accuracy of the significand, while increasing the size of the exponent increases the range of numbers that can be represented . . . good design demands good compromises.").

One of skill in the art would have been motivated to achieve the power-savings identified by Tong and Dockser by performing operations that reduced floating-point precision in a variety of systems, including personal computing devices utilizing a single instruction multiple

datastream architecture. For instance, MacMillan, U.S. Patent No. 5,689,677, discloses achieving performance improvements for "a computer system" by incorporating "a Single Instruction Multiple Data (SIMD) parallel processing capability" that "can be added inexpensively to existing system architectures," including "popular, low cost computer systems running popular operating systems, such as Microsoft Windows based personal computers (including Windows 3.1, Windows NT and Windows 95), Apple compatible personal computers, and UNIX-based systems (including those by Sun Microsystems, Silicon Graphics, Hewlett Packard, and Digital Equipment Corporation)." MacMillan at 5:48-6:10; *see also id.* at 16:14-27 (noting that "a system in accordance with the present invention significantly increases the computing power of computer systems for personal use" and "[t]he low cost nature of this architecture helps generate a high volume of sales, maximizing the architecture's appeal to software developers"). Such a system could include hundreds of processing elements with a 32-bit wide data path, capable of including floating point accelerators, and whose operation is controlled by a CPU:

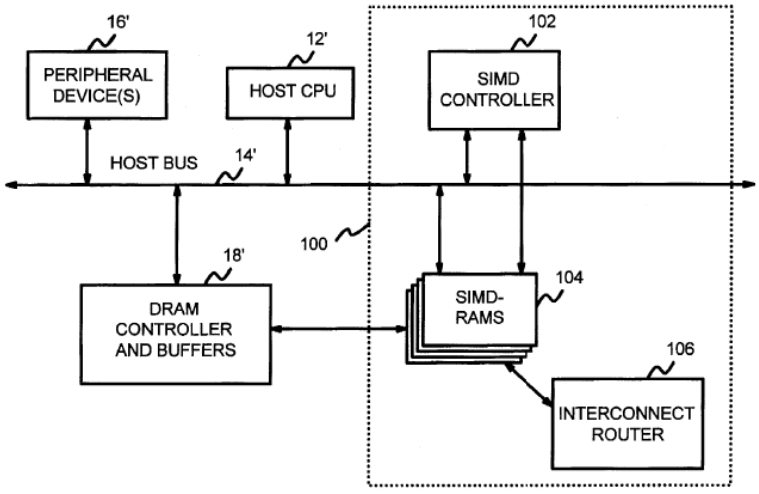


FIG. 2

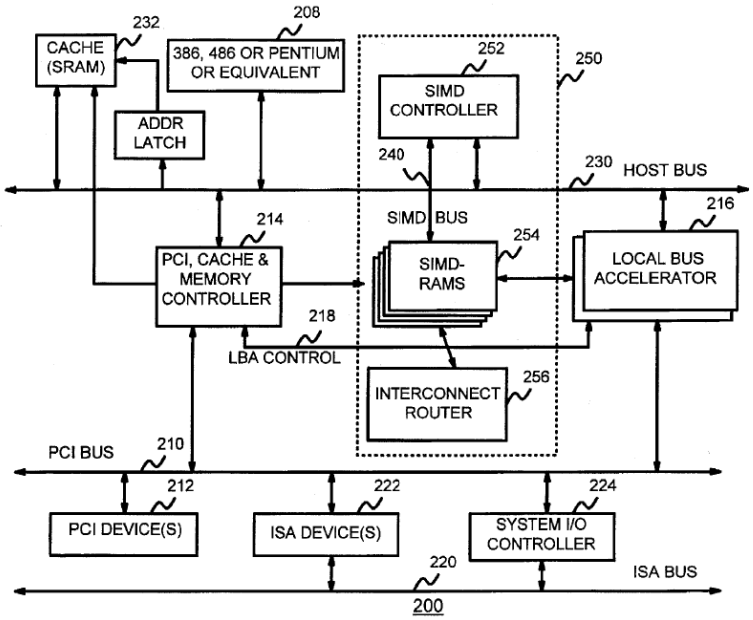

CACHE (SRAM)
232
386, 486 OR PENTIUM OR EQUIVALENT
208
ADDR LATCH
SIMD CONTROLLER
252
250
240
230
HOST BUS
216
214
SIMD BUS
254
PCI, CACHE & MEMORY CONTROLLER
SIMD-RAMS
LOCAL BUS ACCELERATOR
218
LBA CONTROL
INTERCONNECT ROUTER
256
PCI BUS
210
212
222
224
PCI DEVICE(S)
ISA DEVICE(S)
SYSTEM I/O CONTROLLER
220
ISA BUS
200

FIG. 3

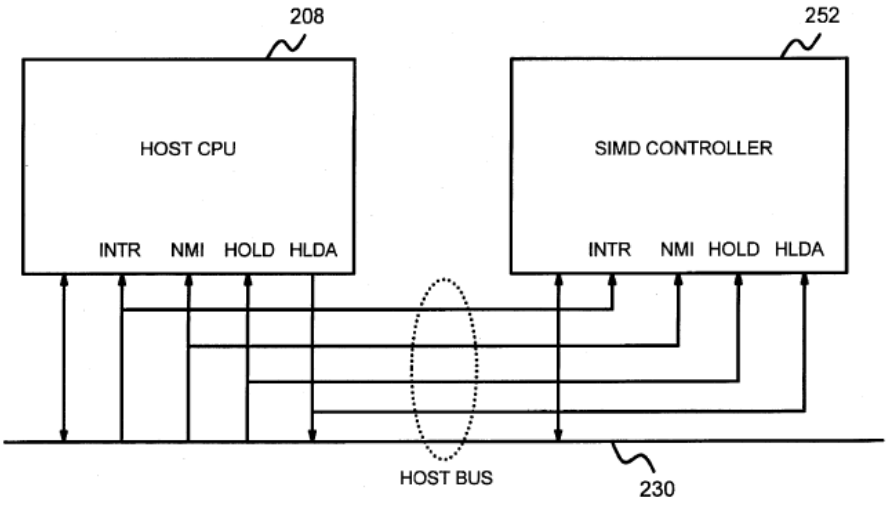

208
252
HOST CPU
SIMD CONTROLLER
INTR NMI HOLD HLDA
INTR NMI HOLD HLDA
HOST BUS
230

FIG.4

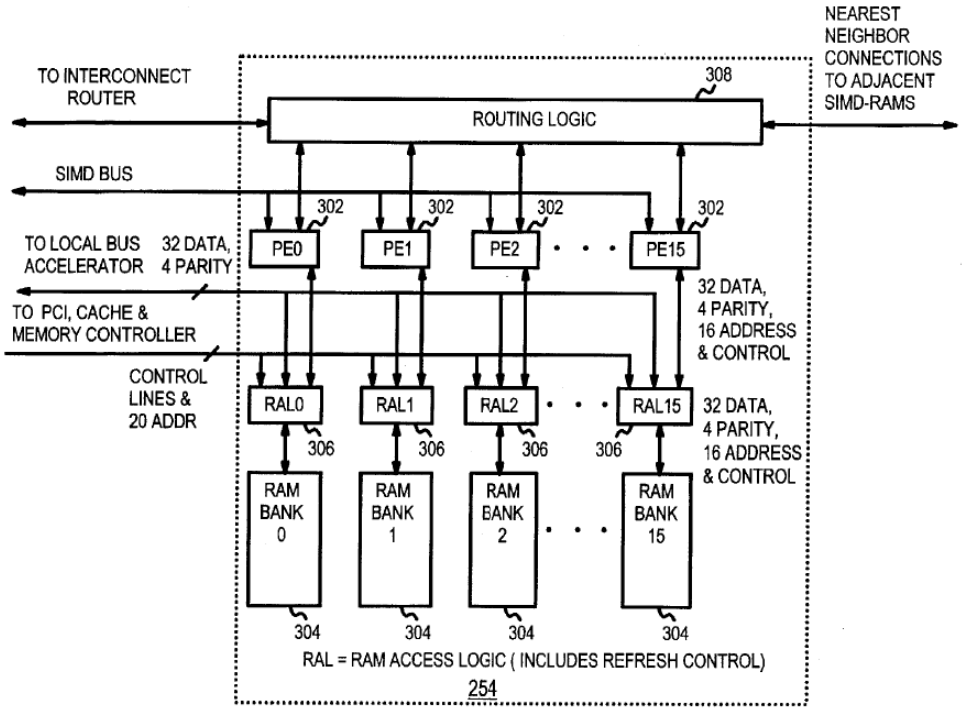


FIG. 5

PENTIUM AND SIMD CONTROLLER ADDRESSES OF:	CORRESPOND TO: PE ADDRESSES	ON PE #
0 THRU 3	0 THRU 3	0
4 THRU 7	0 THRU 3	1
• • •	• • •	• • •
1020 THRU 1023	0 THRU 3	255
1024 THRU 1027	4 THRU 7	0
1028 THRU 1031	4 THRU 7	1

FIG. 6

Id. at Figs. 2-6; *see also id.* at 8:11-20, 9:30-38, 10:24-53, 12:35-13:4, 13:12-13, 13:13-62.

In sum, based on the disclosures identified in Exhibits 6 -7, Dockser, either alone or in combination with the floating point formats disclosed in Tong, Belanović, Belanović and Leeser, Lee, Shirazi, Aty, Sudha, and TMS320C32, would have rendered the asserted claims obvious. To the extent not obvious to one of skill in the art based on common general knowledge, the further combination with MacMillan would have rendered obvious those claims that require that the number of LPHDR execution units in the device exceeds by at least one hundred the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide. In particular, as detailed in Google's petitions for *inter partes* review (incorporated by reference), Dockser either standing alone or in combination with one or both of Tong and/or MacMillan, render the asserted claims obvious.

4. Systems Supporting Variable Precision Floating-Point Numbers and/or Operations

Based on its Infringement Contentions, Singular appears to contend that any device that can perform operations on floating point numbers with a wide dynamic range and that meets the patents' specified error ranges will infringe the patents. The prior art is replete not only with microprocessors but entire systems predicated on such calculations. For instance, no later than 1994, Cray Research, Inc. produced and sold the T3D, a massively parallel supercomputer architecture. A Cray T3D system contained hundreds or even thousands of microprocessors capable of performing arithmetic operations on reduced precision floating-point numbers. Specifically, as detailed in Exhibit 2, the Cray T3D was capable of performing operations on floating-point numbers with a reduced-precision fraction of 5-bits or more and an 11-bit exponent. Thus, the system easily met the dynamic range and imprecision requirements (as interpreted by Singular) of the asserted patents.

Because the system itself was designed to and capable of doing low precision and high dynamic range arithmetic, it anticipates the asserted claims. To the extent that Singular nonetheless contends that one of skill in the art would have needed a motivation to operate the Cray T3D system using the lower-precision formats that the Cray T3D accomodated, one of skill in the art would have been motivated to do so based on the teachings of any of Tong, Dockser, Lee, Belanović, Belanović and Leeser, Shirazi, Aty, Sudha, TMS320C32, Asanović, CNAPS, Synapse-1, SPERT-II, and/or MANTRA I, either alone or in combination.

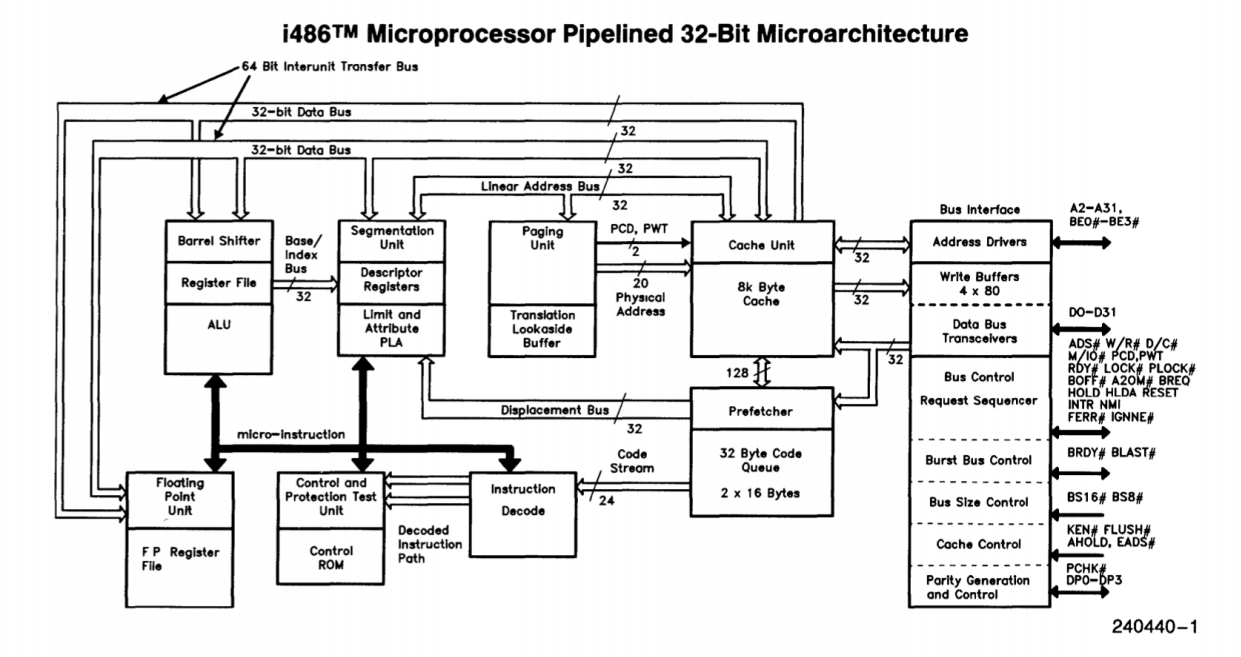
5. Processors Capable of Operating on Reduced-Precision Floating-Point Numbers

As noted above, Singular appears to contend that any device that performs operations on floating point numbers with a wide dynamic range and that results in the patents' specified minimum error ranges infringes the asserted claims. Without elaboration, Singular's Infringement Contentions reproduce a graphic of Google hardware, alongside an illustration of a floating-point format with dedicated exponent bits and a smaller number of mantissa bits. *See* Singular Infringement Contentions, Ex. B at 6. Singular alleges that using this floating-point format, Singular "computed the result and compared it to the result of an exact mathematical calculation," which purportedly demonstrated that Google's products operate within the claimed error ranges. Singular thus appears to contend that the mere input of a relatively low-precision floating-point format—such as a format with 9 or fewer fraction bits and 6 or more exponent bits—satisfies the claim language. In short, Singular's infringement allegations appear to rest on the illogical premise that any processing element performing calculations on certain reduced-precision floating point numbers would be within the scope of the asserted claims.

As noted above, people of skill in the art recognized that reduced-precision operations were well-suited to a wide range of applications, including machine learning. *See generally*

CNAPS, Synapse-1, SPERT-II, MANTRA I, and/or Asanović. Moreover, people of skill in the art knew that reduced precision operations with floating-point-format inputs provided a good balance between precision and dynamic range. *See generally* Tong, Dockser, Lee, Belanović, Belanović and Leeser, Shirazi, Aty, Sudha, and TMS320C32. The prior art is replete with systems on which it would have been feasible and desirable to utilize a low-precision floating-point format as Singular's infringement theories appear to contemplate.

For example, one of skill in the art would have been motivated to use reduced-precision floating-point formats on conventional CPUs such as on an Intel 80486 microprocessor, knowing that the additional precision might be unnecessary:



i486™ MICROPROCESSOR at preamble; *see also id.* at 18 ("The 486 microprocessor is a 32-bit architecture with on-chip memory management, floating point and cache memory units."). Likewise, one of skill in the art would have been motivated to use such reduced-precision floating-point inputs on an Intel system utilizing an Intel 80386 microprocessor. Akin to off-loading floating-point arithmetic to an Intel 80387 co-processor, the system could have off-

loaded tasks requiring lower precision to one or more low precision high dynamic range execution units. *See* Intel 386™ DX Microprocessor Hardware Reference Manual at 1-1 ("The Intel386 DX microprocessor is a 32-bit microprocessor that forms the basis for a high-performance 32-bit system."); *id.* at 1-4 ("The Intel386 DX microprocessor has a numeric coprocessor interface designed for the Intel387 DX math coprocessor. For applications that benefit from high-precision integer and floating-point calculations, the numeric coprocessor provides full support for the IEEE standard for floating-point operations. The Intel387 DX math coprocessor is software compatible with the 80287 and the 8087, earlier numeric coprocessors."). Such implementations could, in turn, be integrated into a wide range of personal computers. *See* Pentium™ Processor User's Manual, Volume 3: Architecture and Programming Manual at preamble ("Intel's 32-bit X86 architecture, represented by the Intel386™ and Intel486™ microprocessor families, is the de facto standard of modem business computing in millions of PCs worldwide.").

Similarly, one of skill in the art would have been motivated to use reduced-precision floating-point numbers in massively parallel processing devices. For instance, one of skill in the art would have been motivated to use such reduced-precision floating-point numbers for Intel's i860 microprocessor, which in turn could be integrated into an iPSC/860 massively parallel supercomputer with over 100 nodes:

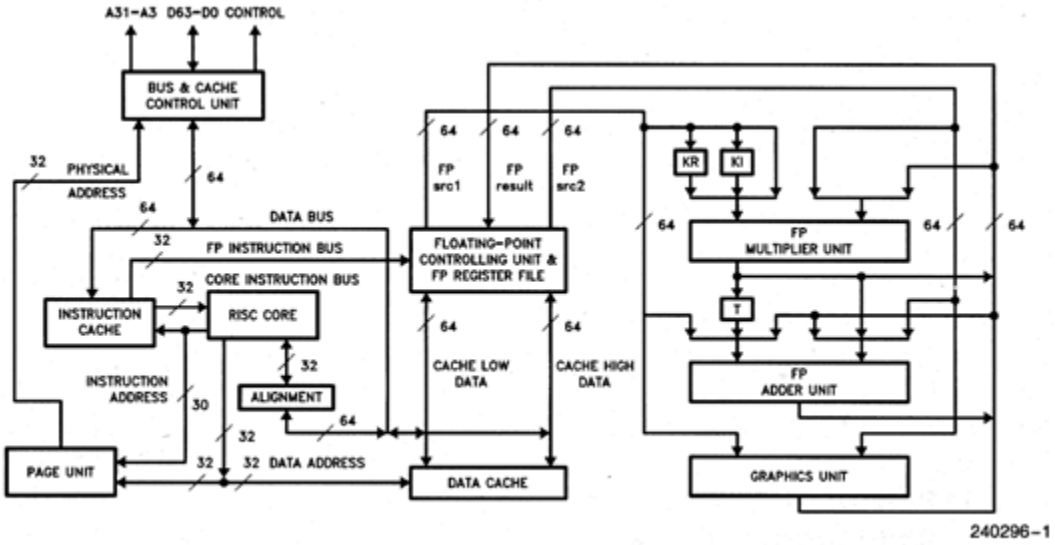


Figure 0.1. Block Diagram

Intel i860™ Microprocessor, at Fig. 0.1; *see also* Barszcz, *One Year with an iPSC/860* at 1 ("The iPSC/860 system is based on the 64 bit microprocessor by Intel. . . . A single node of the iPSC/860 system consists of the i860, eight megabytes (MB) of 70 nanosecond DRAM, and hardware for communication to other nodes. For every 16 nodes, there is also a unit service module to facilitate node diagnostics. The iPSC/860 system at NASA Ames consists of 128 computational nodes."); *id.* at 2 ("The complete [iPSC/860] system is controlled by a system resource module (SRM), which is based on an Intel 80386 processor. This system handles compilation and linking of source programs, as well as loading the executable code into the hypercube nodes and initiating execution.").

As another example, one of skill in the art would also have been motivated to use reduced-precision floating point numbers for a personal computer device utilizing a single instruction multiple datastream architecture. As noted above, MacMillan discloses achieving performance improvements for a computer system by incorporating a SIMD parallel processing capability that could be added to existing system architectures, which could include hundreds of processing elements, capable of including floating point accelerators, and whose operation is

controlled by a CPU. MacMillan at 5:48-6:10, 8:11-20, 9:30-38, 10:24-53, 12:35-13:4, 13:12-13, 13:14-62, 16:14-27, Figs 2-6.

In sum, based on the claim interpretation inherent in Singular's infringement allegations, the asserted claims would have been obvious based on *any* processors that could operate on reduced-precision floating point numbers regardless of the precision level of the processor, either alone or in combination with the reduced precision floating point formats disclosed in Tong, Dockser, Lee, Belanović, Belanović and Leeser, Shirazi, Aty, Sudha, and TMS320C32. *See* Exhibits 6-14, 17.

6. Processors With Reduced-Precision Floating-Point Number Formats

Texas Instruments produced and sold a floating-point digital signal processing (DSP) chip that anticipates and/or renders obvious Singular's asserted claims. Specifically, among other formats (integer and 32-bit floating point), the TMS320C32 used a 16-bit short-form floating-point number format that had 7 fraction bits, 8 exponent bits, and a sign bit—similar to the bfloat16 format accused by Singular in its infringement contentions. The chip was released in approximately 1995, well before the earliest priority dates of any of the asserted patents.

A chart comparing the Asserted Claims to the TMS320C32 chip is attached as Exhibit 14. To the extent that Singular contends that the TMS320C32 does not disclose that "the number of LPHDR execution units in the device exceeds by at least one hundred the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide," that would have been obvious to one of skill in the art. Indeed, later DSPs had multiple cores, and DSPs were often used in parallel for signal processing. In any event, it would have been obvious to one of skill in the art that the processing elements of the TMS320C32 could be deployed in a massively parallel architecture; beyond the basic knowledge of one of skill in the art, examples of this architecture

and disclosed in Belanović, Cloutier, MacMillan and MANTRA I. Along similar lines, and for the reasons explained above in connection with the FPGA prior art, it would have been obvious to use numerous of the signal processing circuits disclosed in the TMS320C32 in parallel. For example, implementing the format used in the TMS320C32 in an FPGA or systolic array would have been obvious given how well known those devices and architectures were to those of skill in the art.

7. Low-Precision ANN Systems Using Fixed-Point Formats

The asserted patents claim an embodiment of Dr. Bates' invention comprising a "massively parallel processor or other device, which includes processing elements designed to perform arithmetic operations . . . on numerical values of low precision but high dynamic range." '273 patent at 2:13-18. But incorporating processing elements designed to perform arithmetic operations of numerical values of low precision and high dynamic range would have been known to a person of reasonable skill in the art as of the priority date(s) of the asserted patents.

The use of processing elements designed to perform relatively low precision calculations in parallel was well-known and well-practiced in the field of artificial neural network ("ANN") design as of the priority date(s) of the asserted patents. At that time, skilled artisans were already designing systems to "try to take advantage of [the] peculiarities of ANN algorithms, such as reduced precision required in the computations," which made it possible to develop "processing elements . . . characterized by small size and cost." Paolo Ienne, *Digital Systems for Neural Networks*, Proc. SPIE 10279, Digital Signal Processing Technology: A Critical Review, April 25, 1995, at 11. That reduced size and cost coupled with the "inherent parallelism in ANN . . . models", in turn, helped "speed up" ANN simulations. Dan Hammerstrom, *Digital VLSI for Neural Networks*, at 2, https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.7541&rep=rep1&type=pdf. In

particular, studies from that period showed that for commonly used ANN algorithms, calculations that used 8- or 16-bit fixed point number formats "perform[ed] as well as [calculations using] 32-bit single-precision floating-point." John Wawrzynek, et al., *Spert-II: A Vector Microprocessor System*, Computer, Vol. 29, No. 3, March 1996, at 80.

That realization led to the creation of numerous ANN systems incorporating parallel processing elements designed to perform calculations that had "typically use[d] 32-bit floating point math" using 8- or 16-bit fixed point math instead. Hal McCartor, *Back Propagation Implementation on the Adaptive Solutions CNAPS Neurocomputer Chip*, Advances in Neural Information Processing Systems 3, 1990, at 1029. These systems included, for example, the CNAPS system developed by Adaptive Solutions, the Synapse-1 system developed by Siemens Munich and the University of Mannheim, and the SPERT systems developed at the University of California at Berkeley and the International Computer Science Institute.

The principal distinction between these systems and the invention claimed in the asserted patents appears to be that the invention in the asserted patents relate to using a floating-point or logarithmic number format whereas the CNAPS, Synapse-1, and SPERT-II systems were designed to perform calculations using a 16-bit fixed point number format. But a person with reasonable skill in the art would have known to adapt these systems to use a floating point number format. "[W]hether to use fixed or floating point calculations is simply a design choice for one skilled in the art." *Tech. for Energy Corp. v. Computational Sys., Inc*., 6 F.3d 788, at *3 n.2 (Fed. Cir. 1993) (adopting conclusion of expert witness). They have many of the same qualities. Like floating point number formats, fixed point formats could be adapted to expand the dynamic range of a calculation at the expense of precision. Jason M. Kinser and Thomas Lindblad, *Implementation of pulse-coupled neural networks in a CNAPS environment*, IEEE

Transactions on Neural Networks, Vol. 10, No. 3, May 1999.

The main difference between the two formats was that the floating-point format provided a much larger possible dynamic range. As a result, certain programs could only run on systems performing floating point math. Ying Fai Tong, Rob A. Rutenbar, and David F. Nagle, *Minimizing Floating-Point Power Dissipation Via Bit-Width Reduction*, Presentation at the Power-Driven Microarchitecture Workshop, June 27-July 1, 1998, at 3, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.5235&rep=rep1&type=pdf. It thus would have been obvious to a skilled artisan to adapt a system like the CNAPS, Synapse-1, or SPERT-II to use floating-point formats. In fact, both the Synapse-1 and SPERT-II systems were adapted to accommodate floating point number formats. U. Ramacher, et al., *SYNAPSE–1: A High–Speed General Purpose Parallel Neurocomputer System*, Proceedings of 9th International Parallel Processing Symposium, Santa Barbara, CA, 1995; J. Wawrzynek, et al., *SPERT-II: a vector microprocessor system and its application to large problems in backpropagation training*, Proceedings of Fifth International Conference on Microelectronics for Neural Networks, Lausanne, Switzerland, 1996, at 624. Accordingly, user reduced-precision floating point formats would have involved relatively basic implementation steps.

Notably, the adoption of fixed-point number formats in ANN systems was largely driven by cost concerns. "[F]loating point units [were] too expensive for massively parallel architectures." Tom Baker & Dan Hammerstrom, *Modifications to Artificial Neural Networks Models for Digital Hardware Implementation*, Oregon Graduate Center, Technical Report No. CS/E 88-035, December 1988, at 6, https://scholararchive.ohsu.edu/downloads/pc289j41v?locale=en. However, that cost difference has become "considerably less pronounced" as "the cost of floating point [chips] has continued

to fall." Gene Frantz & Ray Simar, *Comparing Fixed- and Floating-Point DSPs*, Texas Instruments, 2004, at 2-3, https://www.ti.com/lit/wp/spry061/spry061.pdf?ts=1602718654116&ref_url=https%253A%252F%252Fwww.google.com%252F. And those differences have continued to decrease as demand for floating point units has increased, with the result that the "choice of using a fixed- or floating-point [chip] boils down to whether floating-point math is needed by the application data set." *Id.* Given these reducing costs, a skilled artisan would have known to adapt an ANN system performing calculations using a reduced-precision fixed point number format to perform calculations using a reduced-precision floating point number format to take advantage of the high dynamic range and falling costs of floating point units; moreover, in so doing, it would have been obvious to one of skill in the art to choose one of the floating point formats disclosed in Tong, Dockser, Lee, Belanović, Belanović and Leeser, Shirazi, Aty, Sudha, TMS320C32, either alone or in combination, or the logarithmic formats disclosed in GRAPE-3 and Hoefflinger.

8. Systolic Arrays of Numerous Processing Elements

Dr. Bates did not invent array processors. To the contrary, the common specification distinguishes between vector processors and array processors, both of which were well known in the art by 2008. '273 patent at 3:40-4:6. The common specification also acknowledges that array processors had been built to perform floating-point arithmetic. *Id.* at 3:57-59.

In fact, systolic arrays were well known in the art by 2008. Their invention is often attributed to work by H.T. Kung and Charles Leiserson from the late 1970s to early 1980s. *See, e.g., Systolic Arrays (for VLSI),*"*Proc of Symposium On Sparse Matrix Computations And Their Applications"* (1978); U.S. Patent No. 4,493,048. Systolic arrays were also implemented using floating-point formats in various different WARP machines. *See* H.T. Kung, *Systolic algorithms for the CMU WARP processor* (1984). Notably, the WARP machines (WW-Warp, PC-WARP,

and iWARP) used floating-point numbers. Google's TPUs use such long-known systolic arrays for the MXU, including in the first unaccused version of the TPU. *See* https://cloud.google.com/blog/products/gcp/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu. The architecture is particularly suited to machine learning and using relatively low precision number formats. *See generally* David Zhang and Sankar K. Pal, *Neutral Networks and Systolic Array Design* (2001).

One particular example of a systolic array architecture that used many parallel processing elements for machine learning was the MANTRA I machine. As detailed in Exhibit 1, the MANTRA I system used a systolic array architecture that did reduced precision calculations on fixed point numbers, using only the 16 most significant bits for inference (and not using the remaining bits in the register); however, for the reasons explained in the section immediately above, it would have been obvious to one of skill in the art to use a floating point number format instead, and particularly to use the reduced precision floating point number formats of Tong, Dockser, Lee, Belanović, Belanović and Leeser, Shirazi, Aty, Sudha, and TMS320C32, either alone or in combination.

V. FAILURE TO COMPLY WITH 35 U.S.C. § 112

Google's asserted bases for invalidity under 35 U.S.C. § 112 are based on Google's investigation thus far. Google reserves the right to amend or otherwise modify its asserted bases for invalidity under 35 U.S.C. § 112 based on its review of additional documents and evidence, including, without limitation, additional information concerning the state of the art and level of ordinary skill in the art at the relevant time. Google also reserves the right to assert additional arguments of invalidity under 35 U.S.C. § 112 based on the claim construction process, Singular's proposed claim constructions, Singular's Infringement Contentions, Singular's expert reports, or other positions taken by Singular in this case. A more detailed basis for § 112

defenses will be set forth in Google's claim construction briefing, expert reports, and/or in pleadings.

A. Lack of Written Description Under § 112(1)

Google contends that one or more of the asserted claims are invalid for failure to comply with the written description requirement under 35 U.S.C. § 112(1). The written description, drawings, and claims in a patent must clearly allow a person of ordinary skill in the art to understand and recognize that the patentee invented what is claimed. *Gentry Gallery, Inc. v. Berkline Corp*., 134 F.3d 1473, 1479 (Fed. Cir. 1998). In this regard, the patent must demonstrate by disclosure in the specification to those skilled in the art that the patentee had "possession" of what is now asserted to be the claimed invention. *Vas-Cath Inc. v. Mahurkar*, 935 F.2d 1555, 1561 (Fed. Cir. 1991). Put another way, written description requires an applicant, like Dr. Bates, to demonstrate that he "was in full possession of the claimed subject matter on the application filing date." *TurboCare v. Gen. Elec. Co.*, 264 F.3d 1111, 1118 (Fed. Cir. 2001).

The written description must actually or inherently disclose every claim element. *PowerOasis, Inc. v. T-Mobile USA, Inc.*, 522 F.3d 1299, 1306-07 (Fed. Cir. 2008). It is not enough to say that undisclosed subject matter would have been obvious or within the normal skill set of a person of ordinary skill. *ICU Medical, Inc. v. Alaris Medical Sys., Inc.*, 558 F.3d 1368, 1377 (Fed. Cir. 2009). Instead, the specification must provide sufficient description to show the inventor actually invented what is claimed—a "mere wish or plan for obtaining the claimed invention" is not enough. *Centocor Ortho Biotech, Inc. v. Abbott Labs.*, 636 F.3d 1341, 1348 (Fed. Cir. 2011).

One of ordinary skill in the art at the time of the invention would not have understood Dr. Bates to be in possession of an invention comprising numerous of the limitations of the asserted claims, and the specification does not enable a person of ordinary skill in the art to make and use

such invention without undue experimentation. In particular, the asserted claims recite a "low precision high dynamic range (LPHDR) execution unit adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value." *See* '273 29:65-30:1. But the patents fail to define "execution unit," or even limit it to any specific structure. Rather, as the specification explains, "references herein to 'processing elements' within embodiments of the present invention should be understood more generally as *any kind* of execution unit, whether for performing LPHDR operations or otherwise." *Id.* at 8:8-10. Though the patents claim ranges of valid inputs and recite outputs within particular error ranges, neither the claims nor the specification explains *how* the execution unit produces such results.

The common specification does not describe the full scope of any of the asserted claims. For example, though the specification, when viewed in light of the skill in the art, arguably describes a digital embodiment, it also purports to describe an implementation using analog circuitry. *See* '273 patent, 6:23-50. Indeed, the asserted claims even seek to encompass such an undisclosed analog embodiment by claiming ". . . the statistical mean, over repeated execution of the first operation on each specific input." But the specification does not contain sufficient description to lead a person of ordinary skill in the art to believe that Dr. Bates actually possessed an invention that could be implemented in analog. Thus, a person of ordinary skill in the art would not read the specification to establish possession of the full scope of the invention that Dr. Bates claimed. In fact, Dr. Bates admitted years later that "[he] couldn't figure out how to do it in analog." *See* Practical Approximate Computing at University of California, Berkeley, March 2016, *available at* https://www.youtube.com/watch?v=aHkWX3QctkM. The common specification similarly includes embodiments using any of numerous aspirational technologies

(*e.g.*, DNA computing), but does not describe any claimed execution unit embodied with those aspirational technologies.

B. Lack of Enablement Under § 112(1)

To satisfy the enablement requirement, the specification of a patent must enable a person of skill in the art, as of the filing date, to practice the full scope of the claimed invention without undue experimentation. *Alza Corp. v. Andrx Pharms.*, LLC, 603 F.3d 935, 941-42 (Fed. Cir. 2010); *Sitrick v. Dreamworks, LLC*, 516 F.3d 993, 999 (Fed. Cir. 2008). "Enabling the full scope of each claim is part of the quid pro quo of the patent bargain." *Sitrick*, 516 F.3d at 999 (quoting *AK Steel Corp. v. Sollac*, 344 F.3d 1234, 1244 (Fed. Cir. 2003) (quotations omitted)). If a patent enables some embodiments within the scope of a claim, but not others, then the claim is invalid. *Alza*, 603 F.3d at 939-43 (affirming judgment that claims encompassing medicinal tablets in both osmotic and non-osmotic dosage forms were invalid where specification taught only osmotic dosage forms); *Sitrick*, 516 F.3d at 999-1001 (affirming summary judgment that claims encompassing both video games and movies held invalid where specification only taught use of invention in video games); *Auto. Tech. Int'l, Inc. v. BMW of N. Am., Inc*., 501 F.3d 1274, 1281-85 (Fed. Cir. 2007) (affirming summary judgment that claims encompassing both mechanical and electronic side-impact sensors were invalid where specification taught only mechanical sensors).

The focus of an enablement inquiry is on the teachings in the specification. *See* 35 U.S.C. § 112 ("The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains . . . to make and use the same. . . .") (emphasis added). To satisfy the plain language of 35 U.S.C. § 112, the patentee is "required to provide an adequate enabling disclosure in the specification; it cannot simply rely on the knowledge of a

person of ordinary skill to serve as a substitute for the missing information in the specification." *Alza*, 603 F.3d at 941; *see also Auto. Tech*., 501 F.3d at 1283-84 (claims encompassing electronic sensors held invalid despite the fact that known technologies could be used to create the electronic sensors, where specification did not teach electronic sensors).

The Federal Circuit has identified several factors that a court may consider when deciding whether the specification requires undue experimentation: "(1) the quantity of experimentation necessary, (2) the amount of direction or guidance presented, (3) the presence or absence of working examples, (4) the nature of the invention, (5) the state of the prior art, (6) the relative skill of those in the art, (7) the predictability or unpredictability of the art, and (8) the breadth of the claims." *Alza*, 603 F.3d at 940. The Federal Circuit has made clear, however, that "when there is no disclosure of any specific starting material or of any of the condition under which a process can be carried out, undue experimentation is required." *Id*. at 941 (quoting *Auto. Tech.*, 501 F.3d at 1283-84).

Given the disclosures in the common specification, and as Google understands Singular's claim interpretations, to the extent that the following claim limitations are interpreted to cover the instrumentalities accused in Singular's Infringement Contentions, such limitations of the asserted patent are not enabled, and the claims reciting such limitations are therefore invalid. Again, the common specification does not provide enabling disclosures for the full scope of any asserted claim. As discussed above, though the common specification arguably enables building a digital embodiment of the asserted claims when viewed in light of the person of ordinary skill in the art, it also purports to enable the claimed execution unit implemented using analog circuitry. *See* '971 Patent, 6:20-25. But, again, Dr. Bates later admitted that "[he] couldn't figure out how to do it in analog." *See* Practical Approximate Computing at University of California,

Berkeley, March 2016, available at https://www.youtube.com/watch?v=aHkWX3QctkM. The specification similarly alleges the claimed execution unit could be made using any of numerous aspirational technologies (*e.g.*, DNA computing), but does not describe the design of an execution unit that operates with the claimed imprecision using those aspirational technologies. Thus, a person of ordinary skill in the art would not have been able to practice the full scope of such a claimed invention without undue experimentation.

C. Indefiniteness Under § 112(2)

Google contends that one or more of the asserted claims are invalid as indefinite under 35 U.S.C. § 112(2), which requires that a patent "conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention." This "definiteness standard" requires "clear notice of what is claimed, thereby appris[ing] the public of what is still open to them." *Interval Licensing LLC v. AOL, Inc.*, 766 F.3d 1364, 1370 (Fed. Cir. 2014) (quoting *Nautilus, Inc. v. Biosig Instruments, Inc.*, 572 U.S. 898, 909-910 (2014)). The "statute requires . . . 'that a patent's claims, viewed in light of the specification and prosecution history, inform those skilled in the art about the scope of the invention with reasonable certainty.'" *Id*. (quoting *Nautilus*, 572 U.S. at 910).

Given the disclosure in the specification, and as Google understands Singular's claim interpretations and Infringement Contentions, various claim limitations/terms are indefinite and therefore the asserted claims that recite or incorporate them by dependency are invalid. For example, the asserted claims call for "repeated execution of the first operation on each specific input," but because the claim scope covers an analog embodiment where the result of the operation may change from execution to execution, it fails to indicate to a person of ordinary skill in the art how many times an operation, in the analog embodiment, must be repeated to meet the claim. As another example, the asserted claims use the term "possible valid inputs," but

here, too, Singular's Infringement Contentions appear to render this language at least ambiguous (if not entirely meaningless) by conflating "inputs" with "numerical values." These and potentially other indefiniteness issues may be the subject of claim construction briefing and argument.

Respectfully submitted,

Dated: November 6, 2020

By: */s/ Matthias Kamber*

Gregory F. Corbett (BBO #646394)
gregory.corbett@wolfgreenfield.com
Nathan R. Speed (BBO # 670249)
nathan.speed@wolfgreenfield.com
Elizabeth A. DiMarco (BBO #681921)
elizabeth.dimarco@wolfgreenfield.com
WOLF, GREENFIELD & SACKS, P.C.
600 Atlantic Avenue
Boston, MA 02210
Telephone: (617) 646-8000
Fax: (617) 646-8646

Asim Bhansali (*pro hac vice*)
abhansali@kblfirm.com
KWUN BHANSALI LAZARUS LLP
555 Montgomery Street, Suite 750
San Francisco, CA 94111

Matthias Kamber (*pro hac vice*)
mkamber@keker.com
KEKER, VAN NEST & PETERS LLP
633 Battery Street
San Francisco, CA 94111-1809

Attorneys for Defendant Google LLC

APPENDIX
to Google Responsive Contentions

APPENDIX to Google Responsive Contentions

I. Dynamic Range – Exponent Bits

Floating-point formats that use 5 signed exponent bits, such as the IEEE "half" precision format (aka binary16), can express values in the range of ±65,504, with precision up to 0.0000000596046. *See, e.g.,* "Half-precision floating-point format," *available at* https://en.wikipedia.org/wiki/Half-precision_floating-point_format. Using 6 signed exponent bits increases the dynamic range to a maximum value of $(2 - 2^{-23}) \times 2^{127} \approx 3.4028235 \times 10^{38}$ and a minimum value of $2^{-126} \approx 1.18 \times 10^{-38}$. *See, e.g.,* "Single-precision floating-point format," *available at* https://en.wikipedia.org/wiki/Single-precision_floating-point_format. Thus, any floating-point format using at least 6 signed exponent bits (or a similar scheme such as a 6-bit exponent with a bias value to enable negative-valued exponents) meets the dynamic range requirements of the asserted claims (1/1,000,000 to 1,000,000).

II. Relative Error Calculations

With respect to the prior art reduced precision floating point formats discussed in the Responsive Contentions and related exhibits / prior art charts, the following demonstrates that they disclose the minimum error ranges of the asserted claims. Because multiplier error is independent of the exponent/sign, software tests were run to demonstrate that dropping precision bits from the standard IEEE single-precision format (aka binary32) would result in different minimum error rates, depending on the number of fraction bits used/dropped. Exemplary source code files related to the testing (for certain examples of fraction sizes and error thresholds) are being produced with the responsive contentions: (i) dockser_mult_sp.cu; (ii) dockser_mult.cu; (iii) dockser_sp.cu; (iv) dockser.cu; and (v) trunc.c. An algebraic analysis provides similar proof.

APPENDIX to Google Responsive Contentions

A. Multiplier Relative Error Independent of Exponent/Sign

Given an input pair of floating-point numbers A and B with sign bits S_A and S_B, exponents E_A and E_B, and mantissas M_A and M_B, the exact mathematical calculation of the product $Q = A \times B$ is:

$$Q = (-1)^{(S_A \otimes S_B)} \times 2^{(E_A + E_B)} \times (M_A \times M_B)$$

where '⊗' denotes an XOR operation.

Representing the mantissa product $(M_A \times M_B)$ as V yields:

$$Q = (-1)^{(S_A \otimes S_B)} \times 2^{(E_A + E_B)} \times V$$

Letting V' be the altered (reduced-precision) mantissa product, the product $Q'$of reduced-precision multiplication is:

$$Q' = (-1)^{(S_A \otimes S_B)} \times 2^{(E_A + E_B)} \times V'$$

The claimed relative error percentage Y is:

$$Y = \left| \frac{Q - Q'}{Q} \right| \times 100$$

Substituting the above expressions for Q and Q' yields:

$$Y = \left| \frac{\left((-1)^{(S_A \otimes S_B)} \times 2^{(E_A + E_B)} \times V\right) - \left((-1)^{(S_A \otimes S_B)} \times 2^{(E_A + E_B)} \times V'\right)}{(-1)^{(S_A \otimes S_B)} \times 2^{(E_A + E_B)} \times V} \right| \times 100$$

Factoring out the exponent and sign terms yields:

$$Y = \left| \frac{\left((-1)^{(S_A \otimes S_B)} \times 2^{(E_A + E_B)}\right) \times (V - V')}{((-1)^{(S_A \otimes S_B)} \times 2^{(E_A + E_B)}) \times V} \right| \times 100$$

The exponent and sign terms in the numerator and denominator cancel out, yielding:

$$Y = \left| \frac{V - V'}{V} \right| \times 100$$

Thus, the claimed "Y" percentage produced by performing a reduced-precision multiplication operation depends only on the operand mantissas and is independent of the exponents and signs.

B. Software Demonstration of Dropping Precision Bits

Test code for dropping precision bits from a floating-point number iterates through all possible pairs of normal IEEE-754 single-format floating-point numbers A and B that are non-negative (having '0' sign bit) with zero-valued exponent (such that the mantissa is multiplied by $2^0 = 1$), and for each pair:

1. Performing the exact mathematical calculation (explained below) of $Q = A \times B$;
2. Retaining a specified number of mantissa fraction bits for A and B and zeroing the remaining less-significant fraction bits to create reduced-mantissa operands A' and B';
3. Computing the reduced-precision product $Q' = A' \times B'$; and
4. Computing the relative error percentage $Y = \left|\frac{Q-Q'}{Q}\right| \times 100$.

The program counts the number of input pairs for which Y is at least the minimum percentage recited in a challenged claim.

Testing only non-negative single-format floating-point numbers with zero-valued exponent is sufficient to test all corresponding possible pairs of single-format operands, because the relative error percentage produced by performing a reduced-precision multiplication operation is independent of operand exponent and sign, and the set of all possible single-format operands combines all possible mantissas with all possible exponents and signs. *See supra* II.A.

The results below demonstrate that reducing operand precision produces the claimed minimum relative error percentage Y for more than the claimed minimum percentage X of the possible valid inputs at precision levels retaining the following numbers of fraction bits in the operands for each claimed X/Y combination:

- 9 fraction bits: *see, e.g.*, Dockser,[1] Belanović,[2] Lee, Shirazi, and Aty, Tong, and Cray T3D
- 8 fraction bits: *see, e.g.*, Dockser, Sudha[3], Tong, and Cray T3D
- 7 fraction bits: *see, e.g.*, Dockser, TMS320C32, Tong, and Cray T3D
- 6 fraction bits: *see, e.g.*, Dockser, Tong, and Cray T3D
- 5 fraction bits: *see, e.g.*, Dockser, Tong, and Cray T3D

The reported "X" percentages conservatively exclude pairs that could produce overflow/underflow exceptions.

Retained Fraction Bits	Claimed Y%	X%
9	$\geq 0.05\%$	$\geq 92.14333\%$
	$\geq 0.20\%$	$\geq 14.59791\%$
8	$\geq 0.05\%$	$\geq 97.64554\%$
	$\geq 0.20\%$	$\geq 70.1662\%$
7	$\geq 0.05\%$	$\geq 99.01989\%$
	$\geq 0.20\%$	$\geq 92.11541\%$
6	$\geq 0.05\%$	$\geq 99.36289\%$
	$\geq 0.20\%$	$\geq 97.65571\%$
5	$\geq 0.05\%$	$\geq 99.44834\%$
	$\geq 0.20\%$	$\geq 99.02600\%$

Notably, all of these satisfy the limitations of the asserted claims.

For purposes of this analysis, the "exact mathematical calculation" is of $Q = A \times B$ and can have 48 bits in its mantissa (the product of two 24-bit mantissas). The test software therefore stores the exact product in IEEE-754 double-precision floating-point format. To the extent that

[1] While Dockser expressly discloses an embodiment using 9 fraction bits, it is a result effective variable that could be less. For that reason, it is also listed for the other, lower-precision formats.
[2] This refers to the fraction bits disclosed in Table 2 of Belanović; the Belanović and Leeser system is not limited to this format because the library accommodates using fewer fraction bits.
[3] Sudha discloses using 8 mantissa bits, which may mean using 7 fraction bits or, conservatively, 8 fraction bits.

Singular contends that the "exact mathematical calculation" of multiplication of two single-precision floating-point operands is the product rounded or truncated back to single-precision floating-point format (23-bit-fraction mantissa), the prior-art reduced-precision formats identified above are still met. That is, using this alternative approach to test both ends of the spectrum of the prior art reduced-precision formats—*i.e.*, retaining either 9 fraction bits or 5 fraction bits—produced identical results up to four decimal places of the X percentage.

Retained Fraction Bits	Claimed Y%	X%
9	$\geq 0.05\%$	$\geq 92.14333\%$
	$\geq 0.20\%$	$\geq 14.59792\%$
5	$\geq 0.05\%$	$\geq 99.44834\%$
	$\geq 0.20\%$	$\geq 99.02600\%$

This suggests the same would also be true if retaining 6, 7, or 8 fraction bits.

C. Algebraic Analysis of Dropping Precision Bits

For single-format floating-point operands A and B with sign bits S_A and S_B, exponents E_A and E_B, and mantissas M_A and M_B, respectively, the exact mathematical calculation of multiplying mantissas M_A and M_B is:

$$V = M_A \times M_B$$

Dropping the least-significant bits truncates the mantissa of each operand. Letting M'_A and M'_B be the truncated versions of mantissas M_A and M_B, and V' be the product of M'_A and M'_B that bit-dropping technique produces,

$$V' = M'_A \times M'_B$$

The claimed relative error percentage "Y" is:

$$Y = \left| \frac{V - V'}{V} \right| \times 100$$

See supra Section II.A.

APPENDIX to Google Responsive Contentions

Letting D_A be the difference between M_A and M'_A, and letting D_B be the difference between M_B and M'_B, *i.e.*:

$$M'_A = M_A - D_A$$

$$M'_B = M_B - D_B$$

The output product of the bit-dropping technique is then:

$$V' = M'_A \times M'_B = (M_A - D_A) \times (M_B - D_B)$$

Substituting the above expressions for V and V'into the expression for Y yields:

$$Y = \left| \frac{(M_A \times M_B) - ((M_A - D_A) \times (M_B - D_B))}{(M_A \times M_B)} \right| \times 100\%$$

Expanding the numerator to show all individual products yields:

$$Y = \left| \frac{\big((M_A \times M_B) - (M_A \times M_B) + (D_A \times M_B) + (M_A \times D_B) - (D_A \times D_B)\big)}{(M_A \times M_B)} \right| \times 100\%$$

In the numerator, $(M_A \times M_B) - (M_A \times M_B)$ cancels out, yielding:

$$Y = \left| \frac{(D_A \times M_B) + (M_A \times D_B) - (D_A \times D_B)}{(M_A \times M_B)} \right| \times 100\%$$

The fraction can be re-written as the sum of three fractions:

$$Y = \left| \frac{D_A}{M_A} + \frac{D_B}{M_B} - \frac{(D_A \times D_B)}{(M_A \times M_B)} \right| \times 100\% \qquad (Equation\ A)$$

M_A and M_B are within the range $1 \leq mantissa < 2$ (*see supra* § V.A); therefore $(M_A \times M_B) \geq M_B$. Because D_A and D_B are fractional differences between an original mantissa and its truncated version, D_A and D_B are both less than 1; therefore $(D_A \times D_B) < D_B$. Therefore:

$$\frac{D_B}{M_B} \geq \frac{(D_A \times D_B)}{(M_A \times M_B)}$$

(because the right-hand fraction has a smaller numerator and a larger denominator than the left-hand fraction), and the quantity $\frac{D_B}{M_B} - \frac{(D_A \times D_B)}{(M_A \times M_B)}$ in Equation A is a positive number. Thus, the relative error percentage is at least as large as the first fraction in Equation A; *i.e.*:

$$Y \geq \left|\frac{D_A}{M_A}\right| \times 100\% \qquad (Equation\ B)$$

The ratio $\frac{D_A}{M_A}$ expressed as a percentage therefore provides a lower bound on the relative error percentage resulting from performing a multiplication operation any two reduced-precision operands.

50% of all possible values of M_A have a zero as the first (most-significant) fraction bit (*i.e.*, $M_A = 1.0\ldots$); the other 50% have a one as the first bit (*i.e.*, $M_A = 1.1\ldots$).

The value of D_A (the difference between the full and truncated mantissas of operand A) is determined by the $(K+1)^{th}$ through the 23rd fraction bits of M_A, where K is the number of fraction bits to which M_A is truncated by dropping precision bits; *e.g.*, when K=9, M_A's 10th through 23rd fraction bits are zeroed.

Of the 50% of M_A values that have a first fraction bit of zero, 25% have ones in both the $(K+1)^{th}$ and $(K+2)^{th}$ bits. In this 12.5% of all possible values of M_A (25% of 50%), the value of D_A is at least $\left(2^{-(K+1)} + 2^{-(K+2)}\right)$, which is the difference produced when ones in both the $(K+1)^{th}$ and $(K+2)^{th}$ bits are changed to zeros; and the value of M_A is no larger than $(1.5 - 2^{-23})$, which is the largest possible mantissa having a zero as the first fraction bit (*i.e.*, 1.0111…).

Referring to Equation B, therefore, the following inequality holds for the 12.5% of all possible M_A values with zero in the first fraction bit and ones in the $(K+1)^{th}$ and $(K+2)^{th}$ fraction bits, when truncating the operands to K fraction bits:

$$Y \geq \frac{\left(2^{-(K+1)} + 2^{-(K+2)}\right)}{(1.5 - 2^{-23})} \times 100\% \qquad (Equation\ C)$$

Since each M_A value can be paired with every possible value of M_B, the above Equation C is also true for 12.5% of all possible ***pairs*** of mantissas M_A and M_B; thus, for a given sub-precision

retaining K fraction bits (and dropping/truncating the other fraction bits), over 12% of all possible valid pairs of input operands will produce ***at minimum*** the relative error given by Equation C.

Similarly, 25% of all possible values of M_A have zeros as the first ***two*** fraction bits (*i.e.*, $M_A = 1.00\ldots$), and 25% of those have ones in both the $(K+1)^{th}$ and $(K+2)^{th}$ bits. In this 6.25% of all possible values of M_A (25% of 25%), the value of D_A is at least $\left(2^{-(K+1)} + 2^{-(K+2)}\right)$, and the value of M_A is no larger than $(1.25 - 2^{-23})$, which is the largest possible mantissa having zeros as the first two fraction bits (*i.e.*, 1.00111…); therefore, the following inequality holds for the 6.25% of all possible input pairs in which M_A has zeros in the first two fraction bits and ones in the $(K+1)^{th}$ and $(K+2)^{th}$ fraction bits, when truncating the operands to K fraction bits:

$$Y \geq \frac{\left(2^{-(K+1)} + 2^{-(K+2)}\right)}{(1.25 - 2^{-23})} \times 100\% \qquad (Equation\ D)$$

Thus, for a given sub-precision retaining K fraction bits (and dropping/truncating the other fraction bits), over 6% of all possible valid pairs of input operands will produce at minimum the percent error given by Equation D.

The table below provides the results of evaluating Equations C and D with various values K of retained fraction bits as the selected precision level.

Retained Fraction Bits (K)	**Equation C: Minimum Y for** $X \geq 12\%$	**Equation D: Minimum Y for** $X \geq 6\%$	**Meets X/Y Percentages Recited by Claims:**
9	$\geq 0.0976\%$	$\geq 0.1171\%$	• ’156 patent, claim 7 • ’273 patent, claim 53
8	$\geq 0.1953\%$	$\geq 0.2343\%$	All above
7	$\geq 0.3906\%$	$\geq 0.4687\%$	All above plus ’961 patent, claims 4 and 13

6	$\geq 0.78125\%$	$\geq 0.93750\%$	All above
5	$\geq 1.5625\%$	$\geq 1.8750\%$	All above

D. Logic Bit-Dropping

Bits may also be dropped in the multiplier logic. Testing all possible pairs of non-negative ('0' sign bit) normal IEEE-754 single-format floating-point numbers *A* and *B* with zero-valued exponent (mantissa is multiplied by $2^0 = 1$), and for each pair:

1. Performing the exact mathematical calculation of $Q = A \times B$;
2. Dropping logic bits to compute Q' from *A* and *B*, as explained below; and
3. Computing the relative error percentage $Y = \left|\frac{Q-Q'}{Q}\right| \times 100$.

The program counts the number of input pairs for which Y is at least the minimum percentage recited in a challenged claim.

Testing all possible corresponding pairs of non-negative single-format floating-point numbers with zero-valued exponent suffices for the reasons discussed in Section II.A, *supra*.

The software program computes Q' (step 2 above) using the following example of a technique in which bits are dropped in multiplier logic:

1. Initialize the "output value" of the mantissa product to 0;
2. Interpret the bit sequences of the mantissas of operands A and B (including the implied leftmost "1" bit) as 24-bit integers representing the "multiplicand 402" and "multiplier 404";
3. For each bit in the multiplier,
    a. compute a partial product that equals 0 (when the multiplier bit is 0) or the multiplicand (when the multiplier bit is 1);

b. left-shift the partial product by inserting a number of 0s, at the partial product's right end, equal to the multiplier bit's bit position;

c. Convert to 0 all partial product bits less significant than the "selected subprecision"; and

d. Add the result of step c to the output value.

e. (Repeat step 3 for each bit in the multiplier.)

4. Interpret the generated sum as a binary output number Q' in which the radix point is to the left of the rightmost 46 bits of the generated sum.

The results below, which conservatively exclude pairs that could produce overflow/underflow exceptions from the reported X percentage, demonstrate that dropping bits in the multiplier logic produces the claimed minimum relative error percentage Y for more than the claimed minimum percentage X of the possible valid inputs at precision levels retaining 9 or 5 fraction bits (based on the prior art reduced-precision formats listed above and detailed in the Responsive Contentions and related charts):

Retained Fraction Bits	Claimed Y%	X%
9 fraction bits	$\geq 0.05\%$	$\geq 99.35080\%$
	$\geq 0.20\%$	$\geq 84.99052\%$
8 fraction bits	$\geq 0.05\%$	$\geq 99.44003\%$
	$\geq 0.20\%$	$\geq 98.26561\%$
7 fraction bits	$\geq 0.05\%$	$\geq 99.46230\%$
	$\geq 0.20\%$	$\geq 99.11994\%$
6 fraction bits	$\geq 0.05\%$	$\geq 99.47044\%$
	$\geq 0.20\%$	$\geq 99.37356\%$
5 fraction bits	$\geq 0.05\%$	$\geq 99.47388\%$
	$\geq 0.20\%$	$\geq 99.43546\%$

The program stores the exact product in IEEE-754 double-precision floating-point format. *See supra* Section II.B. But if Singular contends that the "exact mathematical calculation" of multiplying two single-format floating-point operands is a single-format product (23-bit-fraction mantissa), the claims would still be met because tests using this alternative produced results identical to the above table up to five decimal places of the X percentage.

Retained Fraction Bits	Claimed Y%	X%
9 fraction bits	$\geq 0.05\%$	$\geq 99.35080\%$
	$\geq 0.20\%$	$\geq 84.99052\%$
5 fraction bits	$\geq 0.05\%$	$\geq 99.47388\%$
	$\geq 0.20\%$	$\geq 99.43546\%$

Again, this suggests the same would also be true if retaining 6, 7, or 8 fraction bits.

E. Software Demonstration of Narrow Bitwidth Multiplication

To the extent the "LPHDR execution unit" term in the asserted claims is construed to encompass an execution unit that performs multiplication on inputs in a floating-point format that has a smaller number of fraction bits than IEEE single-precision numbers, and that produces an output in the same format ("narrow bitwidth multiplication"), software tests show that the prior art still meets the claims. For the same reasons discussed in Section II.A, *supra*, in narrow-bitwidth multiplication the relative error is independent of the sign and exponent.

Test code for narrow-bitwidth multiplication goes through all possible pairs of normal narrow-bitwidth floating-point numbers A and B. For the purpose of this test, a "normal narrow-bitwidth" floating-point number is a normal floating-point number with a number of mantissa fraction bits equal to some number smaller than the number of mantissa fraction bits in the IEEE-754 single format. The test examines pairs of normal narrow-bitwidth floating-point numbers A

and B that are non-negative (having '0' sign bit) with zero-valued exponent (such that the mantissa is multiplied by 20=1), and for each pair:

1. Perform the exact mathematical calculation (explained below) of Q=A×B;
2. Truncating Q to the bitwidth of the inputs, to create a truncated product Q';
3. Computing the relative error percentage $Y = \left|\frac{Q-Q'}{Q}\right| \times 100$

The program counts the number of input pairs for which Y is at least the minimum percentage recited in a challenged claim.

Testing only non-negative single-format floating-point numbers with zero-valued exponent is sufficient to test all corresponding possible pairs of single-format operands, because the relative error percentage produced by performing a reduced-precision multiplication operation is independent of operand exponent and sign, and the set of all possible single-format operands combines all possible mantissas with all possible exponents and signs. *See supra* Section II.A.

The results below demonstrate that performing narrow-bitwidth multiplication at the following numbers of fraction bits produces the claimed minimum relative error percentage Y for more than the claimed minimum percentage X of the possible valid inputs for some or all of the claimed X/Y combinations in the asserted claims, depending on mantissa size:

- 9 fraction bits: *see, e.g.*, Belanovic, Lee, Shirazi, and Aty
- 8 fraction bits: *see, e.g.*, Sudha
- 7 fraction bits: *see, e.g.*, TMS320C32
- 6 fraction bits: *see, e.g.*, Tong
- 5 fraction bits: *see, e.g.*, Tong

APPENDIX to Google Responsive Contentions

The reported "X" percentages conservatively exclude pairs that could produce overflow/underflow exceptions.

Multiplier Bitwidth	Claimed Y%	X%
9	≥0.05%	≥62.07410%
8	≥0.05%	≥80.03833%
	≥0.20%	≥24.93476%
7	≥0.05%	≥88.46869%
	≥0.20%	≥60.32094%
6	≥0.05%	≥91.24339%
	≥0.20%	≥77.57024%
5	≥0.05%	≥90.24765%
	≥0.20%	≥83.64180%

Nearly all of these formats satisfy all the X/Y combinations recited in the asserted claims.